Nonregular Languages
Summary

- Nonregular Languages
- Prove that certain languages cannot be recognized by any finite automaton
- Pumping Lemma
Regular vs Nonregular Languages

- **Regular languages**
  - Correspond to problems that can be solved with finite memory
    - i.e. finite states

- **Nonregular languages**
  - Correspond to problems that cannot be solved with finite memory
  - May need to remember one of infinitely many symbols
  - Requires infinite memory
Example of a Nonregular Language

- \( L = \{ a^n b^n \mid n \geq 0 \} \)
  - Because of \( n \), we need the same number of a’s and b’s
    - \( \{ \epsilon, ab, aabb, aaabbb, aaaabbbb, \ldots \} \)
  - If \( a^n \) and \( a^m \) (\( n \neq m \)) end up in the same state, \( a^n b^n \) and \( a^m b^n \) end up in the same state
    - DFA will either accept a string not in the language (\( a^m b^n \)) or reject a string in the language (\( a^n b^n \))
    - This means for every \( n \), we need a separate state

- \( n \) is not limited, machine must track unlimited number possible states
  - Finite automata have a finite number of states and can not recognize this language
    - Nonregular Language
Must Prove Infinite Memory is Required

- Languages may not require infinite memory even though it seems so

Example
- \( D = \{ w | w \) has an equal number of occurrences of 01 and 10 as substrings \}\)
  - Seems to require the need for counting occurrences
  - However, can be described by the following regular expression
    - \((1^*0^*1^*)^* \cup (0^*1^*0^*)^*\)
    - \(D\) is a regular language

Easy to prove a language is regular
- Create a finite automata that recognizes it
- Create a regular expression to describe language

Harder to prove a language is nonregular
- Must use other proof methods such as contradictions.
Methods to Prove Irregularity

- Proof by contraction of a property that is required by a regular language

- 3 properties are required for a regular language
  1. Closure of language under regular operations (i.e. union, intersection, complement, star...)
  2. Pumping Lemma
  3. Myhill-Nerode Theorem (won’t be on exams)
     1. Strings $x$ and $y$ are **distinguishable** by language $L$ if some string $z$ exists whereby exactly one of the strings $xz$ or $yz$ belongs to $L$
     2. Let $X$ be a set of strings where every 2 district strings are distinguishable.
     3. Let the index of $L$ be the maximum number of elements in $X$
     4. The theorem states that $L$ is regular iff it has a finite index
        1. In addition, the index is equal to the size of the smallest DFA that recognizes it.
Pumping Lemma

- **Pumping Lemma**
  - If \( A \) is a regular language, then there is a number \( p \) (the pumping length) where if \( s \) is any string in \( A \) of length at least \( p \), then \( s \) may be divided into three pieces, \( s = xyz \), satisfying the following conditions:
    - For each \( i \geq 0, \ xyiz \in A \)
    - \( |y| > 0 \), and
    - \( |xy| \leq p \)

- \( p \) is usually chosen as the number of states in a DFA.
  - If there are no strings in \( A \) that are at least length \( p \), then pumping lemma holds.

- Used to show the irregularity of a language
  - Regular languages always satisfy the pumping lemma
  - Opposite is not true
    - If pumping lemma holds, it does not mean the language is regular
Pumping Lemma Proof

- M is a DFA that recognizes language A.
  - Let \( p = |Q| \) (the number of states in M)
  - Any string at least of length \( p \) can be broken into xyz parts

- Given string \( s \) of at least length \( p \)
  - If \( s \) is length \( n \), it transitions into \( n+1 \) states
  - \( n+1 \) is greater than \( p \)
    - By the pigeonhole principle, some states are repeated
Pumping Lemma Proof

- If $q_9$ is the state that repeats, $s$ can be divided into $xyz$ with respect to $q_9$
  - $x =$ substring before $q_9$, $z =$ substring after $q_9$
  - $y =$ substring between $q_9$ occurrences
    - 2\textsuperscript{nd} condition holds because $|y| \geq 1 > 0$

- This shows that the 1\textsuperscript{st} condition of the pumping lemma is satisfied
  - $xy^iz$ is a string of $A$
    - No matter how many times we use $y$, it will be accepted because of $z$

- By the pigeonhole principle, a repeat must have happened by $p+1$
  - Since $y$ is the repeatable portion, $|xy| \leq p$ (3\textsuperscript{rd} condition)
Pumping Lemma

- All strings longer than the pumping length, $p$, can be “pumped”
  - Contains a section of the string that can be repeated any number of times to create new strings that are a part of the language

- All regular languages have the property stated by the pumping lemma
  - If the language does not have the property, it is nonregular
  - Can be used with proof by contradiction to show that a language is nonregular
Example 1

- Show that $L = \{0^n1^n|n \geq 0\}$ is non-regular using pumping lemma

- Suppose there is a DFA for $L$ with $p$ states

- Find a word $w$ and pump to get a contradiction

- Choose $w = 0^p1^p$
  - Let $w = xyz$ and pump to $xyyz$
  - Contradiction by the following 3 cases
    1. $y$ is all zeros: $xyyz$ has more zeros than ones and does not satisfy $L$'s conditions
    2. $y$ is all ones: $xyyz$ has more ones
    3. $y$ is a mix of ones and zeros: $xyyz$ contains a 1 before a 0 which makes the string not member of $L$
Example 2

• Show that \( L = \{ss \mid s \in \{0,1\}^*\} \) is non-regular using pumping lemma

• Choose \( w = 0^p \ 1 \ 0^p \ 1 \), \( p = \) number of states
  • Because of condition 3 of the pumping lemma, \( |xy| \leq p \)
    • \( xy \) is all zeros
    • Pumping \( y \) makes the string uneven dissatisfying the \( ss \) condition of \( L \)
  • e.g. \( w = 00010001, \ x = 0, \ y = 00, \ z = 10001 \)
    • \( xyyz = 0000010001 \neq ss \)
Example 3: Palindromes

- Show that $L = \{w \in \{0,1\}^* | w = w^{\text{reverse}}\}$ is non-regular using the pumping lemma.

- Choose $w = 0^p \ 1 \ 0^p$
  - Since $|xy| \leq p$, $xy$ is all zeros
  - Since $|y| > 0$, $y$ has at least 1 zero
  - $xyyz$ is not a Palindrome
    - e.g. $w = 0001000$, $x = 00$, $y = 0$, $z = 1000$, $xyyz = 00001000$
Example 4

- Show that $L = \{w \in \{0,1\}^* | w$ contains the same number of zeros and ones$\}$ is non-regular using pumping lemma

- Choose $w = 0^p1^p$
  - Since $|xy| \leq p$, $xy$ is all zeros
  - Since $|y| > 0$, $y$ contains at least 1 zero
  - $xyyz$ does not contain an equal number of ones and zeros
  - e.g. $w = 000111$, $x = 0$, $y = 00$, $z = 111$, $xyyz = 00000111$
Example 5

- Show that \( L = \{1^n | n \text{ is a prime number} \} \) is non-regular using pumping lemma.

  - Choose \( w = 1^n \), with \( n \geq p \).

  - \( w = 1^n = xyz = 1^a1^b1^c \).

  - Pumping \( y \) does not guarantee that \( xy^iz \) will have a prime number of ones.
    - Contradiction.
Example 6: pump down

- Show that $L = \{0^i1^j | i > j\}$ is non-regular by pumping lemma

- Can't pump up since $i > j$

- Choose $w = 0^{p+1}1^p$
  - Since $|xy| < p$, $xy$ is all zeros
  - Since $|y| > 0$, $y$ has at least one zero
  - Removing $y$ will mean $i \leq j$, contradiction
  - e.g. $w = 0000111$, $x = 000$, $y = 0$, $z = 111$
    - $xyyz = 00000111$ is in $L$ but
    - $xz = 00111$ isn't
We can ask general questions about DFAs, NFAs, and regular expressions and try to answer them algorithmically, that is, by procedures that could be programmed in some ordinary programming language.

Represent the DFAs, etc., by strings in some standard way, e.g., tuples with some encoding of a transition table.

Sample questions:
- Acceptance: Does a given DFA $M$ accept a given input string $w$?
- Non-emptiness: Does DFA $M$ accept any strings at all?
- Totality: Does $M$ accept all strings?
- Nonempty Intersection: Do $L(M_1)$ and $L(M_2)$ have any string in common?
- Subset: Is $L(M_1)$ a subset of $L(M_2)$?
- Equivalence: Is $L(M_1)$ equal $L(M_2)$?
- Finiteness: Is $L(M)$ a finite set?
- Optimality: Does $M$ have the smallest number of states for a DFA that recognizes $L(M)$?